



(11)

EP 2 723 093 A1

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
23.04.2014 Bulletin 2014/17

(51) Int Cl.:
H04N 21/443^(2011.01) H04N 21/4627^(2011.01)

(21) Application number: **13004995.0**

(22) Date of filing: **18.10.2013**

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB
GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO
PL PT RO RS SE SI SK SM TR**
Designated Extension States:
BA ME

(30) Priority: **18.10.2012 US 201261715785 P**
13.11.2012 US 201261725964 P
13.11.2012 US 201261725790 P
20.12.2012 US 201213721959

(71) Applicant: **Broadcom Corporation**
Irvine, CA 92617 (US)

(72) Inventors:
• **Kellerman, Marcus**
San Diego, CA 92127 (US)
• **Rajgopal, Narayan**
San Jose, CA 95129 (US)
• **Stults, Joshua**
Irvine, CA 92603 (US)
• **Cernekee, Kevin**
Irvine, CA 92604 (US)

(74) Representative: **Jehle, Volker Armin**
Bosch Jehle Patentanwaltsgesellschaft mbH
Flüggengstrasse 13
80639 München (DE)

(54) **Set top box application in a concurrent dual environment**

(57) A set top box or like device utilizing virtualization techniques to isolate secure device resources from an untrusted software framework incorporated in the device. In one implementation, a first virtual machine container is provided for secure execution of a traditional set top box application, while a second virtual machine container is utilized to host a software framework or untrusted portions of a software framework. A secure access client/server interface is provided to support interactions between the first and second virtual machine containers. The software framework may comprise, for example, an Android framework supported by an underlying Linux operating system environment and isolated in a Linux resource container. Virtual container constructs in various embodiments may employ varying levels of hardware sandboxing, including use of dedicated processing resources in multi-processor environments. In further embodiments, the software framework may be partitioned into trusted and untrusted portions that are executed in separate virtual containers.

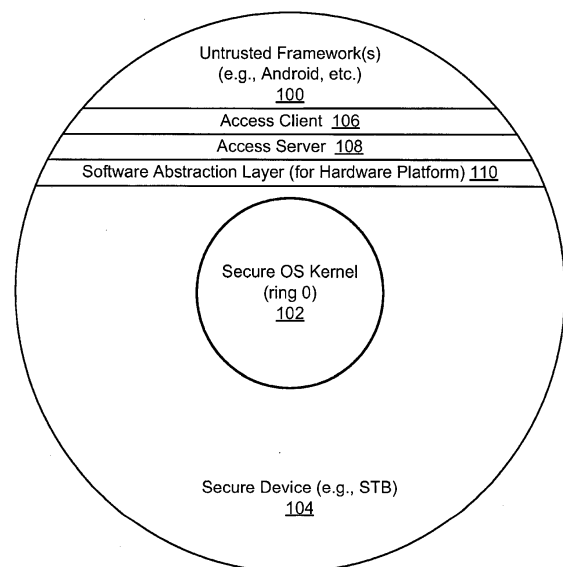


FIG. 1

EP 2 723 093 A1

Description

CROSS REFERENCE TO RELATED PATENTS/PATENT APPLICATIONS

Provisional Priority Claims

[0001] The present U.S. Utility Patent Application claims priority pursuant to 35 U.S.C. § 119(e) to the following U.S. Provisional Patent Applications which are hereby incorporated herein by reference in their entirety and made part of the present U.S. Utility Patent Application for all purposes:

1. U.S. Provisional Patent Application Serial No. 61/715,785, entitled "INTEGRATION OF UNTRUSTED FRAMEWORK COMPONENTS WITH A SECURE OPERATING SYSTEM ENVIRONMENT" (Attorney Docket No. BP30636), filed October 18, 2012, pending.
2. U.S. Provisional Patent Application Serial No. 61/725,964, entitled "INTEGRATION OF UNTRUSTED APPLICATIONS AND FRAMEWORKS WITH A SECURE OPERATING SYSTEM ENVIRONMENT" (Attorney Docket No. BP30637), filed November 13, 2012, pending.
3. U.S. Provisional Patent Application Serial No. 61/725,790, entitled "SET TOP BOX APPLICATION IN A CONCURRENT DUAL ENVIRONMENT" (Attorney Docket No. BP30691), filed November 13, 2012, pending.

BACKGROUND OF THE INVENTION

TECHNICAL FIELD OF THE INVENTION

[0002] The present invention relates generally to processing systems and, more particularly, to the use of virtualization techniques when incorporating untrusted elements in a trusted or secure operating system environment such as a set top box.

DESCRIPTION OF RELATED ART

[0003] Attempts to accommodate relatively untrusted software applications or frameworks in certain types of devices may complicate applicable certification processes that are performed in accordance with one or more industry defined certification standards or testing procedures. For example, vendor certification of many set top boxes used by cable operators is conducted via rigid certification processes that test interoperability and security compliance for devices that implement specifications such as DOCSIS®, PacketCable™, CableHome™, and OpenCable™. Successful certification of such devices typically requires full end-to-end security. Likewise, digital rights management (DRM) and other technologies implemented in a particular device or component (such

as a system-on-a-chip) may require distinct certification processes which do not allow certain interactions with an unsecured operating system (OS) environment or software framework.

[0004] In general, any unsecured portions of a media pathway in a set top box or like device, if not isolated, will cause the certification of such pathway to fail. Most conventional set top boxes are designed to provide a single secure and certified pathway through the various elements of the device. In some recent multi-processor set top boxes, a conventional certified pathway is separated from untrusted components by creating a hardware boundary between the two. In particular, a first processing module is utilized to provide secure functionality (e.g., decoding operations), while a separate processing module with a lower security level is used to support an untrusted framework.

[0005] As is known, a software framework may provide application programming interface functionality and services that are not offered by an underlying operating system, and may thereby offer a level of platform independence in certain implementations. Frameworks are often designed to be a reusable and adaptable software system or subsystem. For example, Android has become one of the fastest-growing operating systems/frameworks for mobile devices. Android, which builds on contributions from the open-source Linux community, provides development tools and reusable components for building applications that can be deployed across many different types of devices, such as a smartphone or tablet device.

[0006] A typical Android framework is a "multi-user" Linux-based system in which each Android application is a different "user" having code that runs in general isolation from other applications. Such process isolation (or application "sandbox") provides a certain level of security. However, various components, processes, threads, etc. used by an application may not entail sufficient protection when integrated in a set top box-type device, resulting in Android being considered an "untrusted" framework when used in such devices. For example, an application that visits an arbitrary web page or receives code from an unverified third party may result in untrusted JavaScript code being executed on a set top box, possibly with elevated privileges. Such code might exploit weakness in other code (e.g., browser code) and receive unauthorized access to file systems, etc., thereby compromising the security of a device, exposing protected data or introducing system instability.

[0007] As indicated above, the processing unit of some devices may have multiple processors or processing cores in order to provide higher performance and/or multitasking capabilities. In some of these multi-processor systems, when multiple applications or programs are running, access control is typically needed to separate the functionality of the applications running on multiple processors. Separation or segregation of different applications and/or tasks running on different processors helps

to ensure that one application does not interfere with the execution of another. Likewise data assigned to one processor should not be accessed by another processor, unless that data is shared between the two processors. Such separation is typically handled through use of virtual memory, with each process having a unique view of memory that is not accessible from outside processes. Hardware access can be handled through a kernel or device driver interface, which provides some level of security. Even in a multi-processor system in which one processor environment provides trusted or secure operations while another operates in an unsecure or restricted environment, however, there can be a substantial possibility of an incursion from the unsecure zone into the secure zone when the operating system is managing the separation.

[0008] For example, in a set top box that allows a user to receive television signals and also allows the user to access the Internet, the secure environment may run applications (including a secure set top box application) pertaining to the reception, decryption and display of certain channels or content provided by a cable or satellite provider or other service operator. The unsecure environment in the set top box may execute applications, such as Android-based applications, that allow a user to access the Internet for web browsing, gaming, etc. In this example, the content provider would generally not want the user or anyone else to access the applications pertaining to broadcast or premium channels. However, if there is commonality in software that controls the accesses to both environments, such as running the same operating system to manage accesses in both environments, then there may be a heightened risk of access violations. Such violations, whether intentional or non-intentional, could result in an unsecure breach into the secure applications of the set top box, such as a web-originated intrusion into protected television channels.

[0009] Accordingly, there is a need to obtain an efficient way to integrate untrusted frameworks, framework components and applications with a secure set top box application.

BRIEF SUMMARY OF THE INVENTION

[0010] According to an aspect, an apparatus operable as a set top box comprises:

processing circuitry;
a set top box application for performing secure media processing operations;
a software framework for supporting relatively untrusted software applications; and
an operating system executed by the processing circuitry, the operating system configured to establish:

a privileged operating system environment to perform secure media processing operations;
and

a restricted operating system environment,

the set top box application executed in the privileged operating system environment and the software framework executed in the restricted operating system environment.

Advantageously, the privileged operating system environment comprises a first virtual machine container and the restricted operating system environment comprises a second virtual machine container.

[0011] Advantageously, the operating system is a symmetric multiprocessing Linux-based operating system.

[0012] Advantageously, the software framework comprises at least one version of an Android framework.

[0013] Advantageously, the operating system further comprising a secure access interface to support interactions between the first and second virtual machine containers.

[0014] Advantageously, the operating system comprising a secure operating system kernel, the secure access interface regulating interactions between the restricted operating system environment and the secure operating system kernel.

[0015] Advantageously, the secure access interface performing at least one operation selected from the group consisting of: handle and pointer validation, non-pointer parameter validation, heap isolation, and release of hardware resources following termination of a restricted process.

[0016] Advantageously, the secure access interface comprising a secure access server and secure access client, the secure access client incorporated in the software framework.

[0017] Advantageously, interactions between the secure access server and secure access client occur, at least in part, through inter-process communication calls.

[0018] According to an aspect, a method is provided for use in a media delivery device having an operating system and trusted application software, the device supporting interaction with an untrusted software framework, comprising:

establishing a privileged operating system environment to perform secure media processing operations by the trusted application software;

establishing a restricted operating system environment to host the untrusted software framework; and
providing a secure access interface to support interactions between the trusted application software and components of the untrusted software framework.

[0019] Advantageously, the privileged operating system environment comprises a first virtual machine container and the restricted operating system environment comprises a second virtual machine container.

[0020] Advantageously, the operating system is a symmetric multiprocessing Linux-based operating system.

[0021] Advantageously, the untrusted software framework comprises at least one version of an Android framework.

[0022] Advantageously, the device further having hardware resources for performing media processing operations, wherein the secure access interface permits relatively greater access to the hardware resources by the first virtual machine container as compared to the second virtual machine container.

[0023] Advantageously, the device having a first processing unit and a second processing unit, wherein the first virtual machine container is executed on the first processing unit and the second virtual machine container is executed on the second processing unit.

[0024] According to an aspect, an apparatus comprises:

processing circuitry;

a software framework to support software applications, the software framework comprising a trusted portion and an untrusted portion; and

an operating system executed by the processing circuitry, the operating system configured to establish at least one privileged operating system environment and at least one restricted operating system environment,

the trusted portion of the software framework executed in the at least one privileged operating system environment and the untrusted portion of the software framework executed in the at least one restricted operating system environment.

[0025] Advantageously, the apparatus further comprises a set top box application, the set top box application executed in the at least one privileged operating system environment.

[0026] Advantageously, the at least one privileged operating system environment comprises a first virtual machine container and the at least one restricted operating system environment comprises a second virtual machine container.

[0027] Advantageously, the software framework comprises at least one version of an Android framework.

[0028] Advantageously, the apparatus further comprises a secure access interface to support interactions between the at least one privileged operating system environment and the at least one restricted operating system environment.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0029]

FIG. 1 illustrates integration of an untrusted framework and secure operating system kernel in accordance with an embodiment of the present disclosure.

FIG. 2 illustrates integration of an untrusted frame-

work as a client of a set top box application in accordance with various embodiments of the present disclosure.

FIG. 3 illustrates concurrent operation of set top box application and an untrusted framework utilizing virtual containers in accordance with various embodiments of the present disclosure.

FIG. 4 illustrates partitioning of a framework into trusted and untrusted portions in accordance with an embodiment of the present disclosure.

FIG. 5 illustrates a set top box architecture utilizing certified applications in accordance with various embodiments of the present disclosure.

FIG. 6 illustrates integration of trusted and untrusted software in a single processor environment in accordance with an embodiment of the present disclosure.

FIG. 7 illustrates integration of trusted and untrusted software in a multi-processor environment in accordance with an embodiment of the present disclosure.

FIG. 8 is a block diagram of a multi-processor system supporting hardware segregation of operating system environments in accordance with various embodiments of the present disclosure.

FIG. 9 is a schematic block diagram of a set top box (STB)/gateway (GW) in accordance with an embodiment of the present disclosure.

FIG. 10 is a logic diagram of a method for supporting interaction between a privileged operating system environment and an untrusted software framework in accordance with an embodiment of the present disclosure.

DETAILED DESCRIPTION OF THE INVENTION

[0030] While certain embodiments are described in conjunction with an Android framework and/or Linux operating system, other embodiments of the present invention may be practiced with a variety of current or future operating systems/kernels and frameworks, and employ a variety of computing circuits, devices, servers and/or systems that might utilize multiple processors, processing cores and/or processing circuits. Further, certain of the illustrations herein describe a processing module, a processor or a CPU (e.g. CPU1, CPU2) for a device that provides a processing function in the described embodiments. However, it is appreciated that a variety of other devices and/or nomenclature may be used in other embodiments to provide for the processing function in practicing the invention. The invention may be readily adapted to other usages where multiple processing environments (zones, domains, etc.) exist, in which separation and/or segregation between two or more zones is desired. Likewise, while certain embodiments are described as implemented by a set top box (STB) or like media processing device performing trusted media processing operations, the novel architectures and methodologies are applicable in whole or in part to other devices, including media

consumption devices such as PVR's, DVD players, access points, televisions, computing devices, smart-phones, etc.

[0031] FIG. 1 illustrates integration of an untrusted framework 100 (such as an Android framework) and secure operating system (OS) kernel 102 within a secure device (such as a certified set top box device) 104 in accordance with an embodiment of the present disclosure. In general, the OS kernel 102 may be viewed as the heart (or ring 0) of the operating system, providing an interface between system hardware and the rest of the operating system and installed applications.

[0032] As described more fully below, a secure software abstraction layer 110 is provided to isolate access to underlying software, hardware and secure components of the device. Communications and data exchanges between the untrusted framework 100 and secure OS kernel 102 may occur via a secure access layer or interface comprised of a secure access client 106 and a secure access server 108 that support an application programming interface (API) and secure inter-process communication (IPC) calls or kernel drivers. In certain embodiments, the secure access client 106 may be part of or accessible by the framework 100, while the secure access server 108 is incorporated in a secure operating system environment or executed as a secure kernel module. In one embodiment, the secure access interface may help prevent system breaches or destabilization resulting from bad data or parameters by performing, without limitation, parameter validation/checking and peripheral range checking through hardware (when applicable), handle and pointer validation, heap isolation, and release of hardware resources following termination of a restricted process.

[0033] FIG. 2 illustrates integration of an untrusted framework 202 as a client of a set top box application ("STB application") 204 in accordance with various embodiments of the present disclosure. As shown, a STB application 204 is provided to perform basic operations of a set top box 200, and interfaces with an untrusted "client" framework such as an Android framework 202. In this embodiment, the STB application 204 is executed in a secure or privileged Linux kernel environment, such as a Symmetric Multiprocessing (SMP) Linux kernel environment 206, that includes a secure access kernel driver 208 and an access server 216 that includes a full access interface 210 for use by the STB application 204, and a more limited guest access interface 212 for use by the Android framework 202 via an access client 218. The STB application 204 includes access client 214 functionality that supports communications with the full access interface 210 and enables the Android framework 202 to run as an untrusted client of the secure Linux kernel environment 206 using, for example, container-based virtualization. As may be appreciated, the client/server architecture allows untrusted processes to crash or terminate without compromising underlying hardware and causing system instability.

[0034] In addition to an access client 218, the Android framework 202 of the illustrated embodiment may include applications and application support 220, JavaScript interpretation, browser functions 222, plug-ins (e.g., an Adobe Flash plug-in), and a Dalvik virtual machine. The Android framework 202 further includes platform or operating system access functionality 224 that enables limited access to necessary components of the secure Linux kernel environment 206 through a minimal guest Linux services access interface 226.

[0035] As noted, Android provides a component-based framework for developing mobile applications. Each application may be comprised of differing numbers and types of components. By way of example, such components can include "Activity" components that form the basis of a user interface, where each screen presented to the user is a different Activity. Background processing is performed by "Service" components, which also define arbitrary interfaces for communicating with other applications. "Content Provider" components function to share information in a relational database form, while "Broadcast Receiver" components operate as an asynchronous mailbox for messages from the system and/or other components. The Android framework 202 of the disclosed embodiment supports varying degrees of collaboration and dependencies between applications.

[0036] However, the interfaces used by applications to provide services and communicate with other applications or an underlying operating system must be secured to defend against breaches and general malfeasance. By default, applications in a typical Android framework, do not have permission to perform operations that might negatively impact other system components (such as reading or writing private data or another application's files, performing network access, etc.). Because Android essentially "sandboxes" applications from each other, applications must explicitly share resources and data by declaring permissions needed for additional capabilities that are outside of the basic sandbox. For example, an application may statically declare the permissions it requires, and at the time of installation the Android framework may prompt a user for appropriate consents. As noted above in the Description of the Related Art, and without rigid hardware isolation and separate execution environments, the potential for execution of unauthorized code from an untrusted framework may compromise overall system security and negatively impact certification procedures for certain devices absent methodologies and architectures (e.g., a secure access server) such as those described herein.

[0037] In the illustrated embodiment, the untrusted Android framework 202 is executed in a first virtual machine (VM) guest Linux container(s) 230 (such as an "LXC" or Linux resource container) to provide further isolation from secure processes and resources. In this embodiment, the STB application may similarly operate in a second VM host Linux container 232 having full host Linux services access 228. Briefly, and in general, container-based

virtualization (or operating system-level virtualization) allows a kernel to run with a plurality of isolated virtual machines or virtual environments installed on top of it. Each virtual environment may be dedicated to run a particular application, and is typically not a complete operating system instance, but rather a partial instance of the operating system that works with a virtualization layer (or hardware resource abstraction layer) in the host operating system kernel.

[0038] Although virtualization may mitigate the need for hardware separation of untrusted components, in certain embodiments multiple processors having different levels of security may benefit from virtual container constructs which operate wholly via one of such processors, while other containers may span multiple processors (see, for example, the embodiment illustrated in **FIG. 7**). In one such embodiment, a first processor(s) may be dedicated to execution of non-secure functionality, while a second, secure processor(s) may be dedicated to conventional STB functionality. In various alternate embodiments, an untrusted framework may be hardware sandboxed via, for example, non-processor hardware/peripherals, memory isolation from processors and/or peripherals, etc. Various other divisions between secure and untrusted software and hardware are contemplated (e.g., dividing an untrusted framework or applications themselves into a "trusted" portion and an "untrusted" portion), examples of which are described below. Further, by utilizing hardware and operating system virtualization features, multiple operating systems may run simultaneously on the same hardware.

[0039] **FIG. 3** illustrates concurrent operation of set top box application 302 and untrusted client applications/frameworks 304 utilizing virtual containers 306 and 308, respectively, in accordance with various embodiments of the present disclosure. In the illustrated embodiment, untrusted client applications and frameworks interact with secure portions of an underlying set top box platform, including the set top box application 302 and secure Linux kernel environment 310, through application IPC calls and an access minimal client 312 that accesses minimal "guest" Linux services access 316. Likewise, applications in a trusted user mode (or "privileged") operating system environment receive full access to "host" Linux services access 318 via an access client 314. The trusted user mode environment can be supported by a full host root filesystem 320, while a minimal guest root filesystem 322 on a separate partition or disk can support the untrusted user mode (or "restricted") operating system environment.

[0040] In some embodiments, certain resources of a set top box platform, such as media processing and hardware acceleration resources (for audio/video encoding and decoding, rendering of 2D and 3D graphics using a standardized API (such as Open Graphics Library or "OpenGL"), DRM, encryption/decryption, etc.) and networking interfaces, may be accessed by untrusted client applications/frameworks 304 through IPC calls commu-

nicated through network sockets. Likewise, events such as IR control signals may be communicated from a set top box platform to untrusted client applications/frameworks 304 through IPC calls (such as a call into an Android API or button).

[0041] In the illustrated embodiment, a framework "aggregator" 324 may be employed to embrace and extend operation with a wide variety of frameworks. In this manner, for example, a service provider can provide secure access to system resources to third party frameworks and other "middleware" offerings. Such offerings may include, by way of example and without limitation, Adobe Air, Adobe Flash, Apple iOS, Microsoft Silverlight, Java applets, and like side-by-side technologies. The framework aggregator 324 may include a set of tailored operating system libraries and HTML capabilities to support such technologies, as well as user interface web engine access, etc.

[0042] In certain embodiments, user interfaces supported by the framework aggregator 324 may enable a user to launch applications supported by an Android framework. For example, the framework aggregator 324 may offer a series of "buttons", including an Android button that provides a visual display of available Android applications. Further, the Android framework and framework aggregator 324 may share a graphics screen. In this manner, it may not be necessary to port functionality into the Android framework that might otherwise be necessary. It is noted that in a given device, an Android framework may be built using a different tool chain than that utilized by other frameworks supported by the framework aggregator 324.

[0043] **FIG. 4** illustrates partitioning of a framework into trusted and untrusted portions 400 and 402 in accordance with an alternate embodiment of the present disclosure. In this embodiment, a trusted portion(s) 400 of a (Android) framework 400, web browser, etc., and a corresponding untrusted portion(s) 402 are executed in one or more separate virtual machine containers 404 and 408, respectively. The trusted portion 400 may include a secure platform and operating system access interface 410. In addition to the trusted portion 400, a set top box application 406 may operate in the virtual machine container 404, or in a separate and secure virtual machine container(s).

[0044] Using an example involving a web browser, the portion of the browser that executes potentially unsafe Flash or JavaScript code (such as a rendering engine) can be executed in the "untrusted" virtual machine container 408 with limited file system and peripheral access, while the portion of the browser that has access to underlying platform hardware and sensitive operating system resources can be executed in one or more "trusted" virtual machine containers 404. In further embodiments, the trusted portion 400 may comprise a secure clone of a framework, or modified version of a framework that supports secure applications and/or secure portions of applications. In such embodiments, unsecure applica-

tions or portions thereof may be executed by portions of a framework residing in an untrusted or restricted user domain or container. As discussed above, various approaches to hardware sandboxing may be employed to further isolate untrusted portions of a framework and to support restricted operating system domains or environments.

[0045] The secure Linux kernel environment 412 of this embodiment includes a secure access kernel driver 414 and an access server 420 that provides a full access interface 416 to support trusted user mode functionality and a guest access interface 418 to support untrusted user mode functionality. Full host Linux services access 422 and minimal guest Linux services access 424 are provided to the trusted and untrusted user mode environments, respectively. As above, the trusted user mode environment can be supported by a full host root filesystem 426, while a minimal guest root filesystem 428 on a separate partition or disk supports the untrusted user mode environment.

[0046] FIG. 5 illustrates a set top box or other secure architecture 500 utilizing certified applications 502 in accordance with the present disclosure. In the illustrated embodiments, the certified applications 502 may be pre-installed or downloaded, for example, from a catalog of certified applications provided by a service operator or cloud-based source. Such applications might include, by way of example, media streaming applications (e.g., Netflix), gaming applications, web browsing applications, iOS-based applications, Android applications, etc. In some embodiments, it may be necessary to incorporate or download additional operating systems to support corresponding classes of certified applications. Alternatively, different certified versions of a given application may be made available to support a variety of installed operating systems and/or frameworks 510. Certified applications 502 may be executed in one or more virtual machine containers 508. Other illustrated elements of FIG. 5 function as described above.

[0047] In one exemplary embodiment, the primary set top box application 504 is constructed as an Android application executed in a virtual machine container(s) 506. Depending on the capabilities of the relevant Android framework, non-standard API's or libraries may be required to support certain set top box functionality, such as PVR, picture-in-picture, transcoding, and channel tuning operations. Further, this approach may require special interfacing (e.g., passing a string to an interface to create an encode path as part of a hardware call) or provision of non-standard functions (or even applications) to address services and functions that may be lacking in a relevant framework or required for certification. For example, an Android framework might be extended to include non-standard functionality -- without negatively impacting compatibility -- by providing services that are "hidden" under the Android porting layers and receive and interpret uniform resource identifiers ("URIs") passed through such layers.

[0048] In other embodiments, instead of an end-to-end security definition being managed by the source or being defined by the content itself (or associated metadata), each application that is downloaded (or preinstalled) includes a full security definition. For example, each application may have its own downloadable conditional access security (dCAS) or downloadable DRM element/definition. A corresponding certification process might entail, for example, certifying a part of an overall secure pathway, with one or more applications providing the final portion of the certification. Alternatively, a pre-defined plurality of conditional access mechanisms may be provided, with a (downloaded) application defining which mechanism is to be used. It may then be possible to certify a secure "boundary" without regard to the conditional access requirements of a particular company or media.

[0049] In various embodiments, a set top box or like device may exchange capabilities with other devices in a media consumption network (e.g., PVR's, DVD players, access points, televisions, computing devices, smartphones, etc.) to compare against the requirements of an application. Once this is done, the application may select which one or more modes of operation, if any, that it will permit. If the set top box cannot support such modes (due to resource competition or otherwise), such modes become unavailable. If available a set of modes of operation can be selected and dynamically switched to other of such modes on the fly during a media stream delivery, which may require coordination with the application or the source to support the transition (possibly requiring quality changes in the media stream).

[0050] FIG. 6 illustrates integration of trusted and untrusted software in a single processor (CPU1) environment in accordance with an embodiment of the present disclosure. In this embodiment, an SMP Linux kernel 600 functions to control hardware and lock guest virtual machines 602 (supporting a "guest" instance of Linux and running untrusted applications) and host virtual machines 604 (running trusted application core services) to one or more separate CPU threads based on performance and/or security isolation requirements.

[0051] For example, the SMP Linux kernel 600 can perform load balancing for processes in both virtual machines across multiple CPU threads or cores when maximum performance is desired. Further, a host virtual machine 604 with control of a guest virtual machine 602 can terminate and restart a process in the guest virtual machine 602 (or the entire container) if it misbehaves or is compromised. Hard limits can be placed on CPU utilization by a guest virtual machine in order to prevent disruption of core underlying device functionality and to facilitate termination of runaway processes. In operation, separate threads may have segregated and/or restricted access to an I/D cache 606 and L2 cache 608.

[0052] FIG. 7 illustrates integration of trusted and untrusted software in a multi-processor system in accordance with an embodiment of the present disclosure. The

illustrated system may be implemented in a device, module, board, etc. One or more components of the system may also be implemented on an integrated circuit chip or on multiple integrated circuit chips. The particular embodiment of **FIG. 7** shows two processing modules identified as CPU1 and CPU2, respectively. Although two processing modules are shown, other embodiments may have more than two processing modules or processors, and it is noted that CPU1 and CPU2 may be comprised of various processing devices, circuitry, etc. In one embodiment, CPU1 and CPU2 may be comprised of different processing cores of a single CPU, or some other processing circuitry.

[0053] Referring more particularly to the figure, the SMP Linux kernel 700 functions to control hardware and lock a guest virtual machine(s) 702 instance of Linux (running untrusted applications) and a host virtual machine(s) 704 (running trusted application core services) to one or more separate threads 1-4 based on performance and/or security isolation requirements. For example, an untrusted application might be executed in a virtual machine container that spans both CPU1 and CPU2. As described more fully below, various portions of I/D caches 706 and 708 and L2 cache 710 may be restricted to a particular thread or CPU, or shared with appropriate safeguards.

[0054] **FIG. 8** is a block diagram of another multi-processor system supporting hardware segregation of operating system environments in accordance with various embodiments of the present disclosure. For example, a first processor(s) CPU1 may be dedicated to conventional STB functionality and support a secure access server 800, while a second processor(s) (web) CPU2 may be dedicated to execution of non-secure functionality and untrusted applications (e.g., an Android framework client 802 or web browser). A trusted execution environment supported by CPU1 can limit access to certain core STB functionality, including hardware acceleration blocks 804, DRM capabilities 806, and the like.

[0055] In one embodiment, CPU1 and CPU2 are both segregated into separate and distinct zones when in a "sandboxing" mode. In another embodiment, the trusted CPU1 is set up having its own segregated regions of memory and also given access rights over some or all address ranges of memory mapped portions of CPU2. Generally, when operating in separate or segregated zones, environments or domains, the two CPUs operate on different applications, so that CPU1 executes one set of instructions, while CPU2 executes a different set of instructions. Segregation or separation of this nature is typically referred to as sandboxing or sandbox mode. The purpose of most sandboxing is to prevent one zone from accessing functionality in the other zone or to have controlled access of one zone into another. In some instances, both zones may be limited from having access to the other zone or only have controlled access between zones. In some applications, one zone may be regarded as a secure, privileged or trusted zone and the other as a restricted, non-secure or non-trusted zone, in which

access by the applications operating on the non-secure zone are prevented or controlled from accessing certain applications running in the secure zone.

[0056] As noted above, a number of devices utilize multiple processors or processing cores to run separate programs, applications, etc. In a situation where one zone is not to have access to a second zone, one way to ensure this separation is by checking the accesses to the system memory. That is, by ensuring accesses that are allocated to CPU1 are not accessed by CPU2, unless the location of the access is a shared location, applications running on CPU2 may be prevented from breaching the functional separation. One way to achieve this protection is to provide an access check and access control to ensure that the correct processing module is accessing a permitted location for that processing module. In the illustrated embodiment, CPU1 and CPU2 may communicate directly through a network socket, IPC calls, etc.

[0057] **FIG. 9** is a schematic block diagram of a set top box (STB)/gateway (GW) 901 in accordance with an embodiment of the present disclosure. The STB/gateway 901 provides a number of functions, including conversion of signals from external sources into content that can be consumed by network devices. The STB/gateway 901 may further operate as a gateway that supports unidirectional or bidirectional communications and bridging between network devices.

[0058] The STB/gateway 901 of the illustrated embodiment interacts with a residential network infrastructure 905 and external media systems 907 via one or more wired and wireless networks/links. The wired and wireless networks/links may utilize one or more of various transmission media - such as coaxial cable, shielded twisted pair cable, fiber-optic cable, power line wires, and wireless media (radio frequencies, microwave, satellite, infrared, etc.) - and operate in accordance with a variety of communication and networking protocols (TCP/IP, UPnP, IPv6, etc.). In addition, the wired and wireless networks/links may comprise a multi-hop network utilizing a spanning tree protocol, direct wireless connections, peer-to-peer links, etc.

[0059] The external media systems 907 may comprise, for example, one or more of cable, satellite and/or terrestrial television systems. Various headend equipment and services can be utilized by these systems, such as a cable headend that receives television signals for further processing and distribution, and may offer various other services such as internet connectivity and VoIP services.

[0060] The STB/gateway 901 of the illustrated embodiment includes a broadcast/unicast/multicast front end 913 that operates to receive uncompressed or compressed digital video, digital audio and other data signals, from either the external media systems 907 or residential network infrastructure 905, for further processing and distribution. The front end 913 comprises tuner circuitry 919a operable to isolate particular channels. Signals from the tuner circuitry 919a are then provided to analog-

to-digital (ADC) circuitry 920a and demodulation circuitry 921a for conversion into binary format/stream. Once in binary format, forward error correction (FEC) circuitry 922a checks the integrity of the received binary stream. Audio, video, and data extracted from the binary stream may then be decoded (e.g., by decoding 925) into formats suitable for consumption by downstream devices. It is noted that demodulation circuitry 921a may support one or more modulation techniques, such as Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation (QAM), Coded Orthogonal Frequency-Division Multiplexing (COFDM), etc.

[0061] The front end 913 may be integrated into one or more semiconductor devices that may further support, for example, interactive digital television, networked DVR functionality, IP video over DOCSIS applications, and 3D graphics support. In addition, multiple tuner circuitry 919a (including in-band and out of band tuners), ADC circuitry 920a and demodulation circuitry 921a may be provided for different modulation schemes and television standards (such as PAL, NTSC, ATSC, SECAM, DVB-C, DVB-T(2), DVB-H, ISDB, T-DMB, Open Cable).

[0062] In one alternative embodiment of the disclosure, functionality of the STB/gateway 901 is performed by a smartphone or mobile computing device. In this embodiment, the "front end" 913 comprises one or more wireless interfaces (including PHY and baseband functions), such as a cellular (3G, 4G, IMT-Advanced, etc.) or wide area network (HetNet, Wi-Fi, WiMax, etc.) interface. The interface may support one or more modulation and multiplexing techniques, such as OFDM, OFDMA, SC-FDMA, QPSK, QAM, 64QAM, CSMA, MIMO, etc. In the illustrated embodiment, the wireless interface comprises a transceiver 919b, analog-to digital (ADC) and digital-to-analog (DAC) circuitry 920b, demodulation and modulation circuitry 921b and FEC (such as turbo codes or LDPC codes) circuitry 922b. Encoding, decoding and transcoding 925 functions may be provided by processing circuitry and storage 911.

[0063] The STB/gateway 901 also includes (wide area network) interface circuitry 915 for communicating with residential network infrastructure 905 and/or external media system 907. Through the communication interface circuitry 915, the STB/gateway 901 may communicate directly with upstream resources, or offer (bidirectional) bridged communications between such resources and devices (e.g., devices 941-949) coupled to the STB/gateway 901.

[0064] In the embodiment of **FIG. 9**, STB/gateway 901 interacts with a variety of devices 941 - 949 via communication interface circuitry 917. For example, a television or display interface module 931 communicates with a (digital) television 941 or other media display device to relay television programming and enable available interactive media services. In certain embodiments, the television or display interface module 931 might include a remote user interface (RUI) server. Similarly, an audio interface 933 provides audio programming or audio li-

brary access to an audio system 943.

[0065] The communication interface circuitry 917 further comprises a remote control interface 935 for receiving control signals from a remote control 945. In addition to traditional remote control operations, the remote control 945 may further offer voice and/or gesture control signals that are relayed or mapped to relevant consumer devices. User interfaces 937 are also provided for communications with one or more user interface devices 947. Gaming interfaces 939 function to provide interactive communications with a gaming system 949. Such communications may involve, for example, online, multiplayer gaming between members of a social network and/or external players in a gaming platform. Power management interface 940 functionality is provided to enable power saving operations between devices 941-949.

[0066] The STB/gateway 901 of the illustrated embodiment includes processing circuitry, operating system(s) and storage 911 (components of which may be comprised of hardware, software, or combinations thereof), services support 923, and decoding/encoding/transcoding functionality 925 to support network interactions such as those described above. Services support 923 in this embodiment includes various functions such as power management 927, bridging 928, and media server-receiver services 929. Other traditional features of a STB/gateway may also be included. For example, the processing circuitry 911 may include a system-on-a-chip or like device(s) that provide a core processor, audio/video decoder, media processor/DSP, graphics core, encryption/decryption core, adaptive transcoding, etc.

[0067] **FIG. 10** is a logic diagram of a method 1000 for supporting interaction between a privileged operating system environment and an untrusted software framework of a media delivery device in accordance with an embodiment of the present disclosure. In step 1002 of the illustrated embodiment, a privileged operating system environment is established to perform secure media processing operations and related functions, such as those performed by a hosted certified set top box application. Next, in step 1004, a restricted operating system environment is established to host and support an untrusted software framework, such as an Android framework. In one embodiment, both the privileged and restricted operating system environments are constructed using virtual machine containers. In step 1006, a secure access interface or layer (such as described above) is provide to support and regulate interactions between components of the privileged operating system environment and components of the restricted operating system environment. The secure access interface may function, for example, to limit access to underlying device resources controlled by a set top box application or other trusted software.

[0068] As may be used herein, the terms "substantially" and "approximately" provides an industry-accepted tolerance for its corresponding term and/or relativity between items. Such an industry-accepted tolerance rang-

es from less than one percent to fifty percent and corresponds to, but is not limited to, component values, integrated circuit process variations, temperature variations, rise and fall times, and/or thermal noise. Such relativity between items ranges from a difference of a few percent to magnitude differences. As may also be used herein, the term(s) "operably coupled to", "coupled to", and/or "coupling" includes direct coupling between items and/or indirect coupling between items via an intervening item (e.g., an item includes, but is not limited to, a component, an element, a circuit, and/or a module) where, for indirect coupling, the intervening item does not modify the information of a signal but may adjust its current level, voltage level, and/or power level. As may further be used herein, inferred coupling (i.e., where one element is coupled to another element by inference) includes direct and indirect coupling between two items in the same manner as "coupled to". As may even further be used herein, the term "operable to" or "operably coupled to" indicates that an item includes one or more of power connections, input(s), output(s), etc., to perform, when activated, one or more its corresponding functions and may further include inferred coupling to one or more other items. As may still further be used herein, the term "associated with", includes direct and/or indirect coupling of separate items and/or one item being embedded within another item. As may be used herein, the term "compares favorably", indicates that a comparison between two or more items, signals, etc., provides a desired relationship. For example, when the desired relationship is that signal 1 has a greater magnitude than signal 2, a favorable comparison may be achieved when the magnitude of signal 1 is greater than that of signal 2 or when the magnitude of signal 2 is less than that of signal 1.

[0069] As may also be used herein, the terms "processing module", "module", "processing circuit", and/or "processing unit" may be a single processing device or a plurality of processing devices. Such a processing device may be a microprocessor, micro-controller, digital signal processor, microcomputer, central processing unit, field programmable gate array, programmable logic device, state machine, logic circuitry, analog circuitry, digital circuitry, and/or any device that manipulates signals (analog and/or digital) based on hard coding of the circuitry and/or operational instructions. The processing module, module, processing circuit, and/or processing unit may have an associated memory and/or an integrated memory element, which may be a single memory device, a plurality of memory devices, and/or embedded circuitry of the processing module, module, processing circuit, and/or processing unit. Such a memory device may be a read-only memory, random access memory, volatile memory, non-volatile memory, static memory, dynamic memory, flash memory, cache memory, and/or any device that stores digital information. Note that if the processing module, module, processing circuit, and/or processing unit includes more than one processing device, the processing devices may be centrally located

(e.g., directly coupled together via a wired and/or wireless bus structure) or may be distributedly located (e.g., cloud computing via indirect coupling via a local area network and/or a wide area network). Further note that if the processing module, module, processing circuit, and/or processing unit implements one or more of its functions via a state machine, analog circuitry, digital circuitry, and/or logic circuitry, the memory and/or memory element storing the corresponding operational instructions may be embedded within, or external to, the circuitry comprising the state machine, analog circuitry, digital circuitry, and/or logic circuitry. Still further note that, the memory element may store, and the processing module, module, processing circuit, and/or processing unit executes, hard coded and/or operational instructions corresponding to at least some of the steps and/or functions illustrated in one or more of the Figures. Such a memory device or memory element can be included in an article of manufacture.

[0070] The present invention has been described above with the aid of method steps illustrating the performance of specified functions and relationships thereof. The boundaries and sequence of these functional building blocks and method steps have been arbitrarily defined herein for convenience of description. Alternate boundaries and sequences can be defined so long as the specified functions and relationships are appropriately performed. Any such alternate boundaries or sequences are thus within the scope and spirit of the claimed invention. Further, the boundaries of these functional building blocks have been arbitrarily defined for convenience of description. Alternate boundaries could be defined as long as the certain significant functions are appropriately performed. Similarly, flow diagram blocks may also have been arbitrarily defined herein to illustrate certain significant functionality. To the extent used, the flow diagram block boundaries and sequence could have been defined otherwise and still perform the certain significant functionality. Such alternate definitions of both functional building blocks and flow diagram blocks and sequences are thus within the scope and spirit of the claimed invention. One of average skill in the art will also recognize that the functional building blocks, and other illustrative blocks, modules and components herein, can be implemented as illustrated or by discrete components, application specific integrated circuits, processors executing appropriate software and the like or any combination thereof.

[0071] The present invention may have also been described, at least in part, in terms of one or more embodiments. An embodiment of the present invention is used herein to illustrate the present invention, an aspect thereof, a feature thereof, a concept thereof, and/or an example thereof. A physical embodiment of an apparatus, an article of manufacture, a machine, and/or of a process that embodies the present invention may include one or more of the aspects, features, concepts, examples, etc. described with reference to one or more of the embodi-

ments discussed herein. Further, from figure to figure, the embodiments may incorporate the same or similarly named functions, steps, modules, etc. that may use the same or different reference numbers and, as such, the functions, steps, modules, etc. may be the same or similar functions, steps, modules, etc. or different ones.

[0072] Unless specifically stated to the contra, signals to, from, and/or between elements in a figure of any of the figures presented herein may be analog or digital, continuous time or discrete time, and single-ended or differential. For instance, if a signal path is shown as a single-ended path, it also represents a differential signal path. Similarly, if a signal path is shown as a differential path, it also represents a single-ended signal path. While one or more particular architectures are described herein, other architectures can likewise be implemented that use one or more data buses not expressly shown, direct connectivity between elements, and/or indirect coupling between other elements as recognized by one of average skill in the art.

[0073] The term "module" is used in the description of the various embodiments of the present invention. A module includes a functional block that is implemented via hardware to perform one or module functions such as the processing of one or more input signals to produce one or more output signals. The hardware that implements the module may itself operate in conjunction software, and/or firmware. As used herein, a module may contain one or more sub-modules that themselves are modules.

[0074] While particular combinations of various functions and features of the present invention have been expressly described herein, other combinations of these features and functions are likewise possible. The present invention is not limited by the particular examples disclosed herein and expressly incorporates these other combinations.

Claims

1. An apparatus operable as a set top box, comprising:

processing circuitry;
a set top box application for performing secure media processing operations;
a software framework for supporting relatively untrusted software applications; and
an operating system executed by the processing circuitry, the operating system configured to establish:

a privileged operating system environment to perform secure media processing operations; and
a restricted operating system environment,

the set top box application executed in the priv-

ileged operating system environment and the software framework executed in the restricted operating system environment.

2. The apparatus of claim 1, wherein the privileged operating system environment comprises a first virtual machine container and the restricted operating system environment comprises a second virtual machine container.

3. The apparatus of claim 1 or 2, wherein the operating system is a symmetric multiprocessing Linux-based operating system.

4. The apparatus of any preceding claim, wherein the software framework comprises at least one version of an Android framework.

5. The apparatus of any of claims 2 to 4, the operating system further comprising a secure access interface to support interactions between the first and second virtual machine containers.

6. The apparatus of claim 5, the operating system comprising a secure operating system kernel, the secure access interface regulating interactions between the restricted operating system environment and the secure operating system kernel.

7. The apparatus of claim 5 or 6, the secure access interface performing at least one operation selected from the group consisting of: handle and pointer validation, non-pointer parameter validation, heap isolation, and release of hardware resources following termination of a restricted process.

8. The apparatus of any of claims 5 to 7, the secure access interface comprising a secure access server and secure access client, the secure access client incorporated in the software framework.

9. The apparatus of claim 8, wherein interactions between the secure access server and secure access client occur, at least in part, through inter-process communication calls.

10. A method for use in a media delivery device having an operating system and trusted application software, the device supporting interaction with an untrusted software framework, comprising:

establishing a privileged operating system environment to perform secure media processing operations by the trusted application software;
establishing a restricted operating system environment to host the untrusted software framework; and
providing a secure access interface to support

interactions between the trusted application software and components of the untrusted software framework.

11. The method of claim 10, wherein the privileged operating system environment comprises a first virtual machine container and the restricted operating system environment comprises a second virtual machine container. 5
12. The method of claim 10 or 11, wherein the operating system is a symmetric multiprocessing Linux-based operating system. 10
13. The method of any of claims 10 to 12, wherein the untrusted software framework comprises at least one version of an Android framework. 15
14. The method of any of claims 11 to 13, the device further having hardware resources for performing media processing operations, wherein the secure access interface permits relatively greater access to the hardware resources by the first virtual machine container as compared to the second virtual machine container. 20 25
15. An apparatus, comprising:
processing circuitry;
a software framework to support software applications, the software framework comprising a trusted portion and an untrusted portion; and
an operating system executed by the processing circuitry, the operating system configured to establish at least one privileged operating system environment and at least one restricted operating system environment,
the trusted portion of the software framework executed in the at least one privileged operating system environment and the untrusted portion of the software framework executed in the at least one restricted operating system environment. 30 35 40

45

50

55

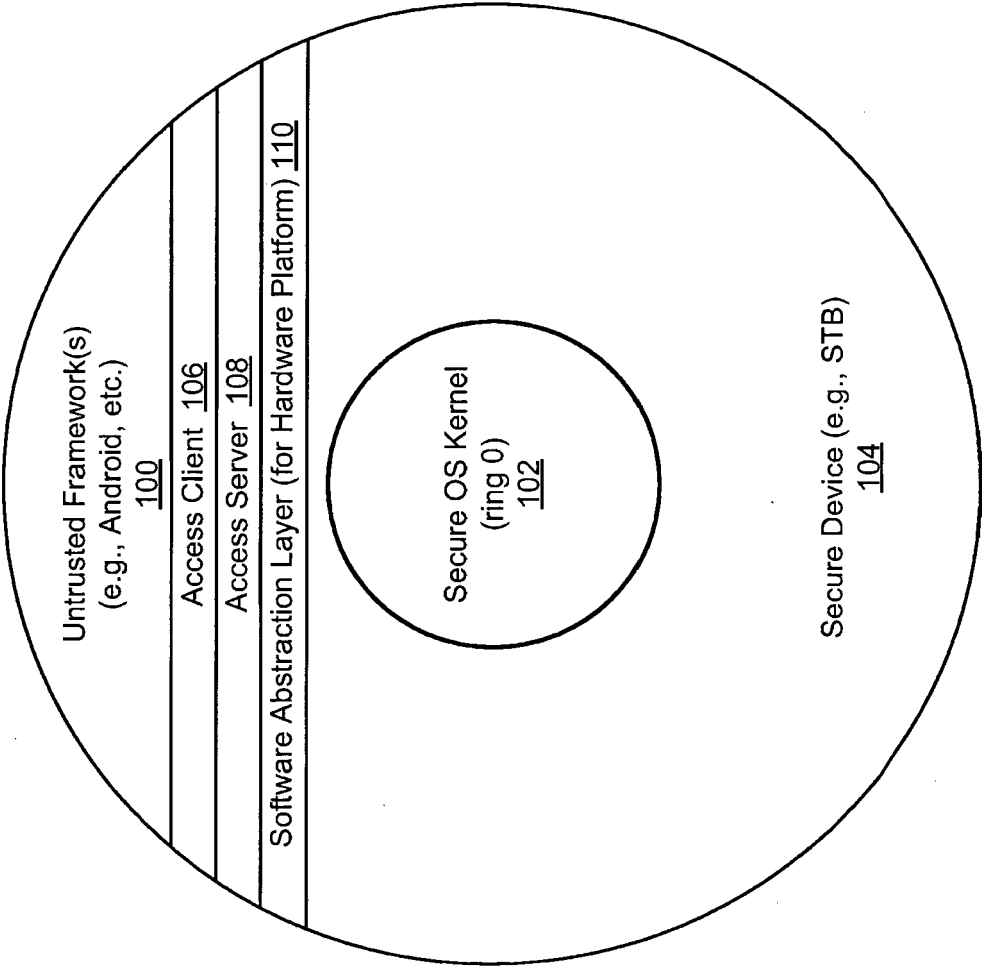


FIG. 1

200

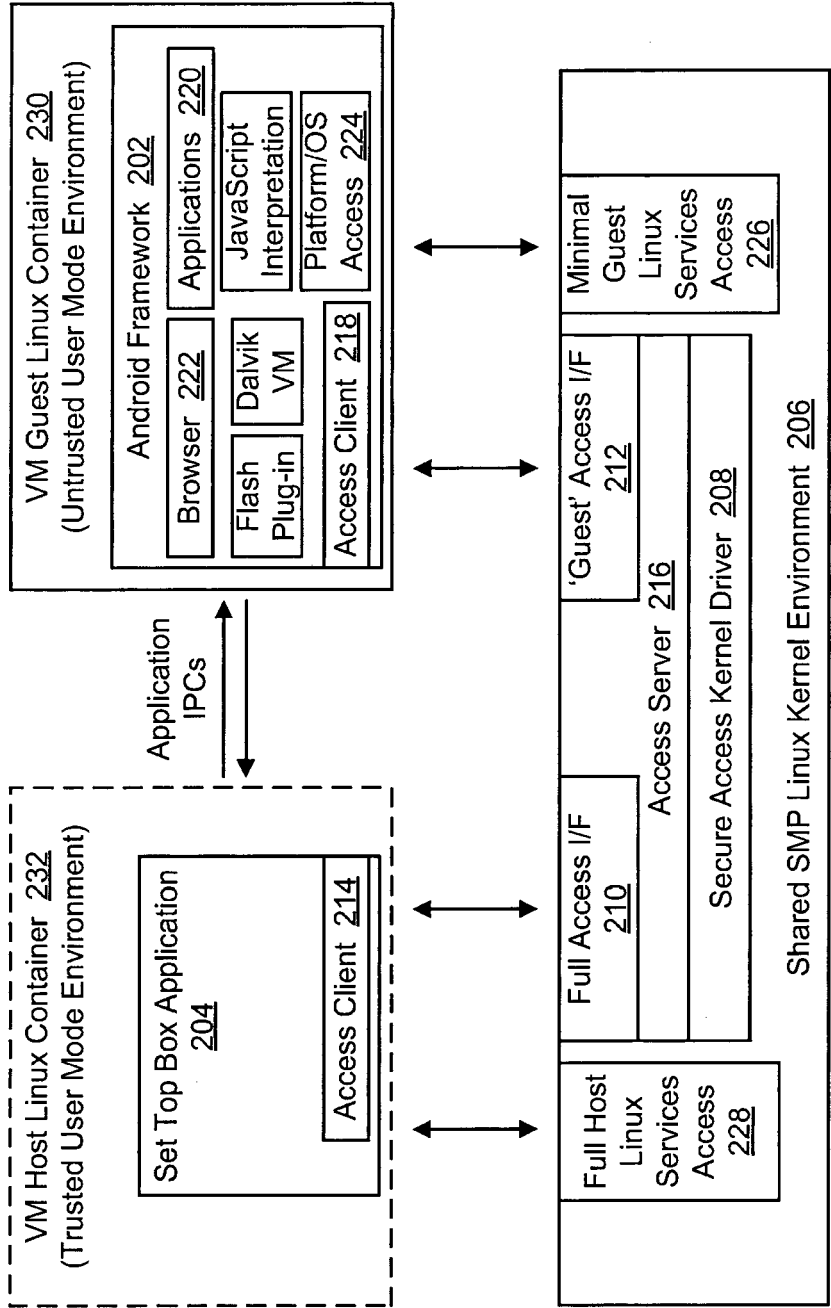


FIG. 2

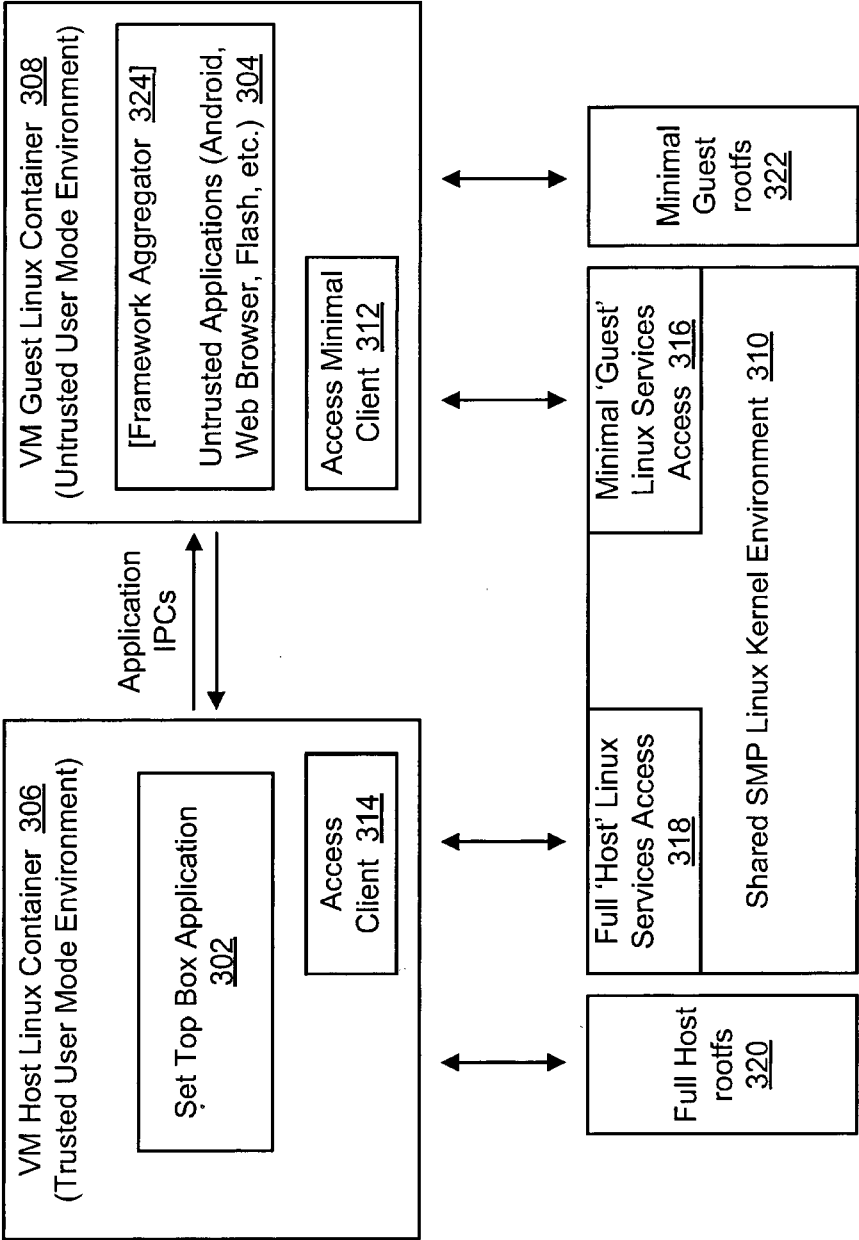


FIG. 3

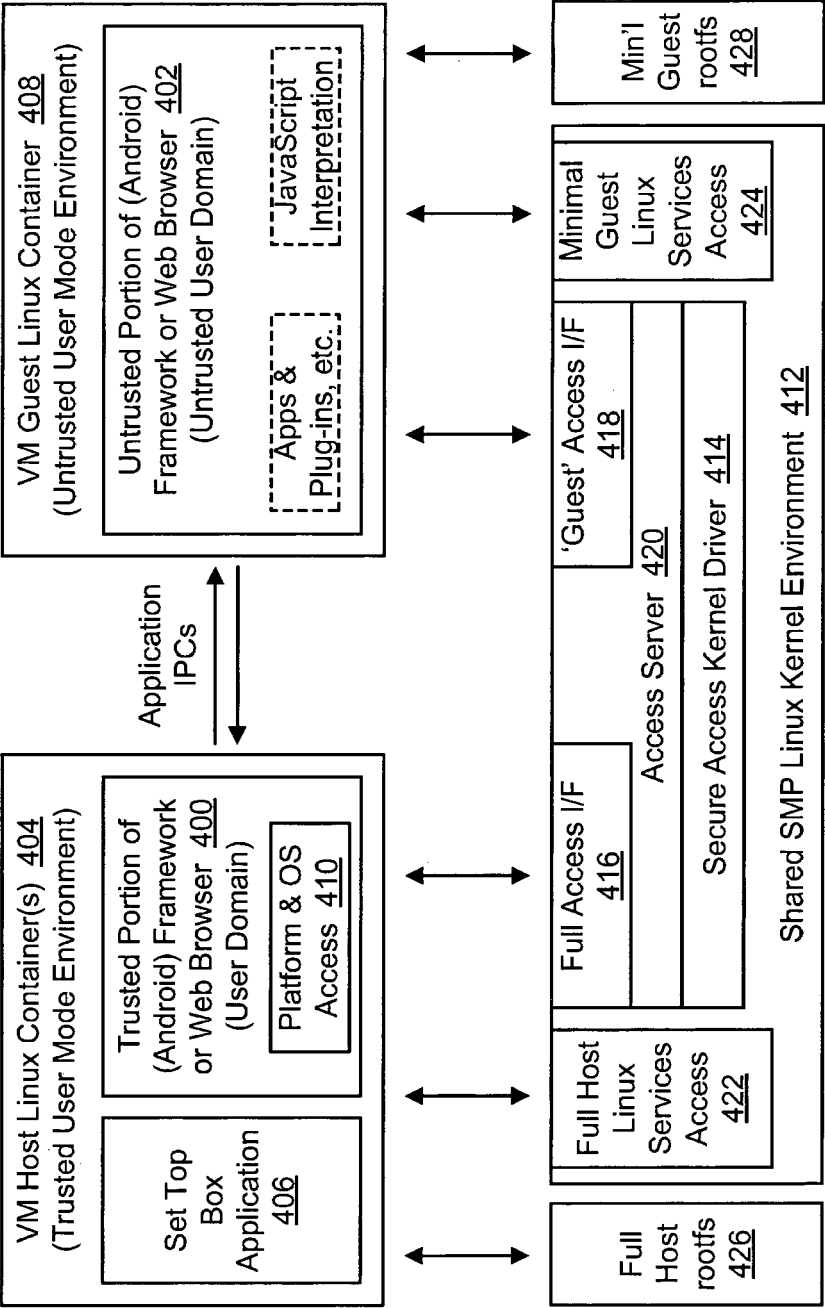


FIG. 4

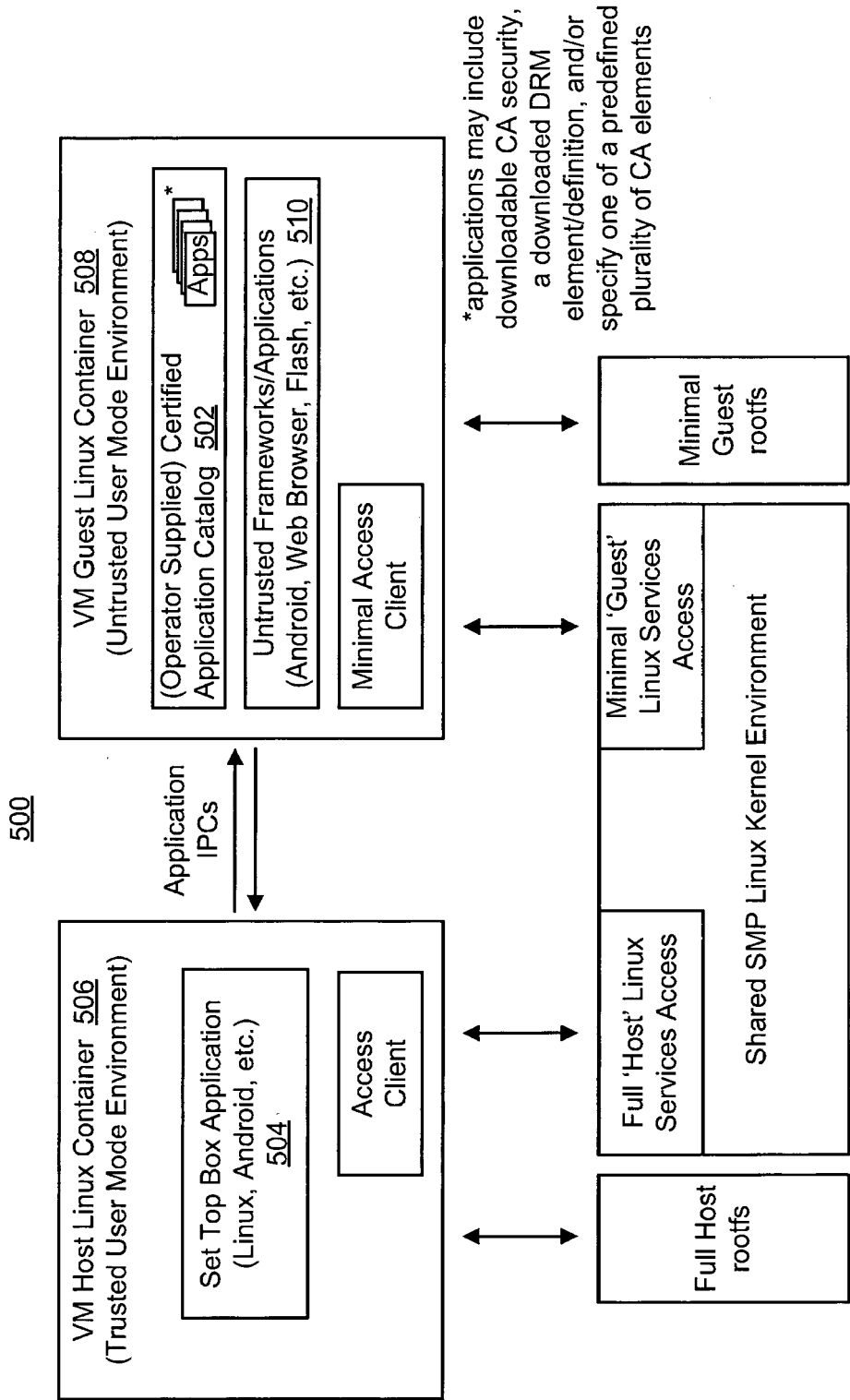


FIG. 5

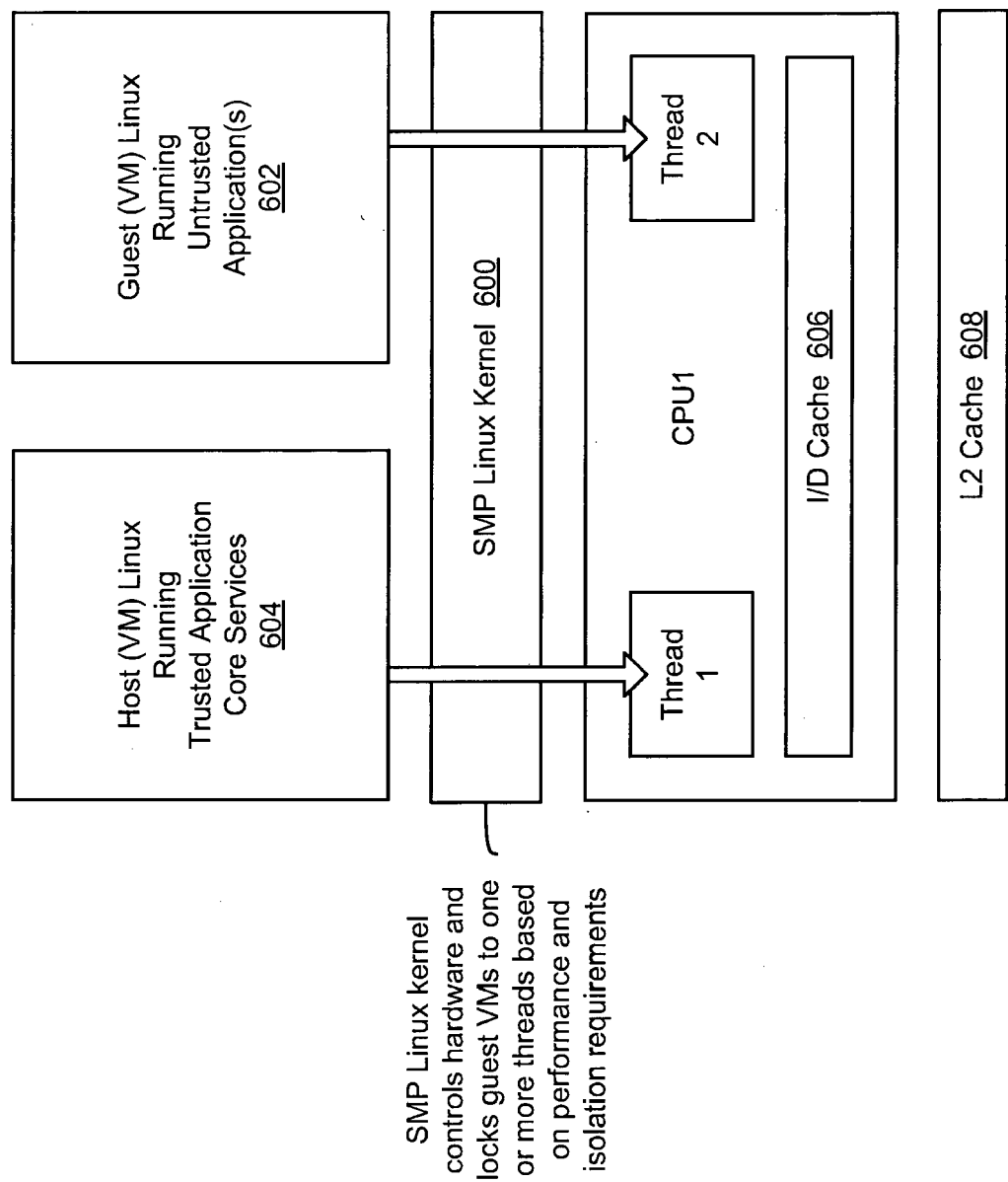


FIG. 6

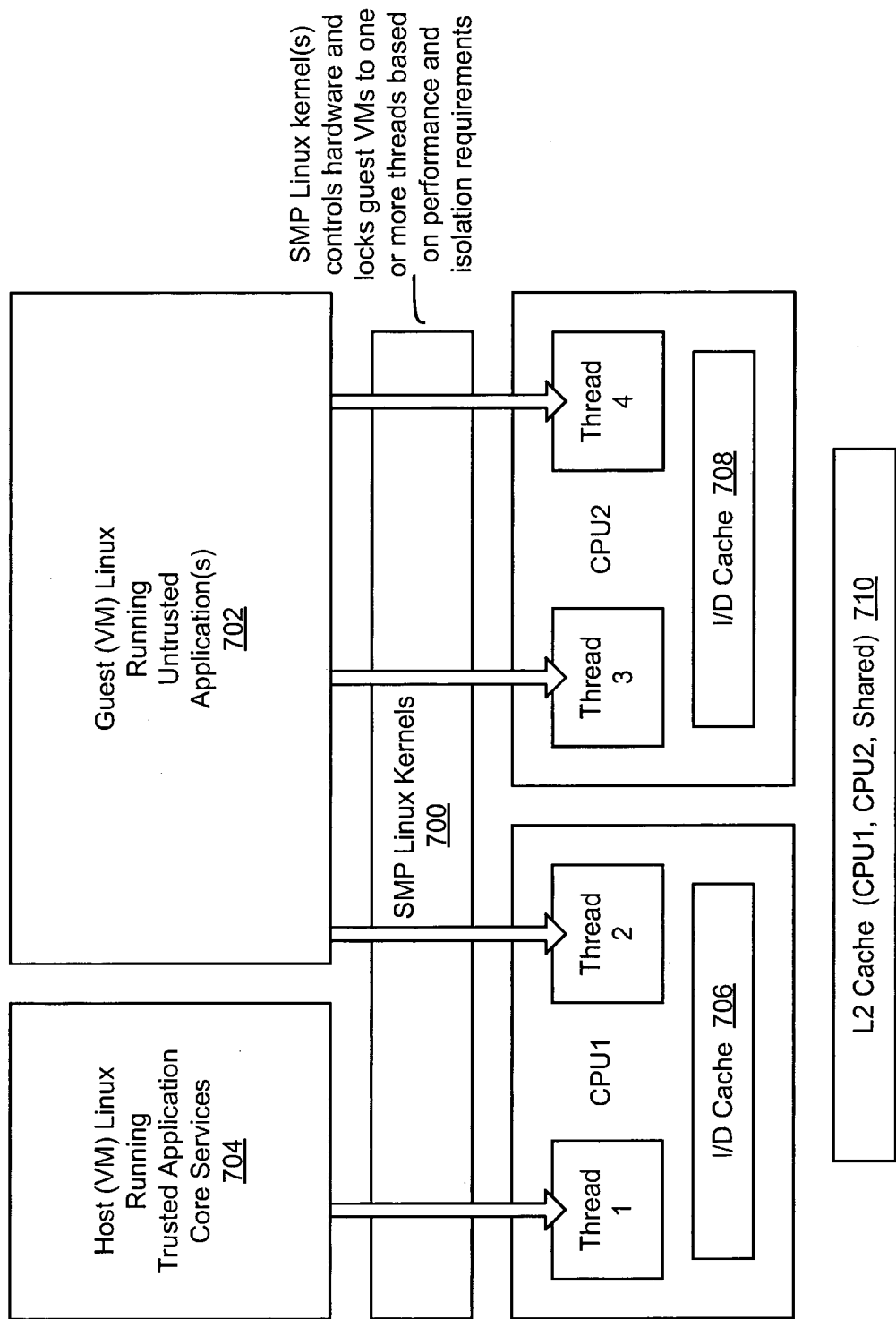


FIG. 7

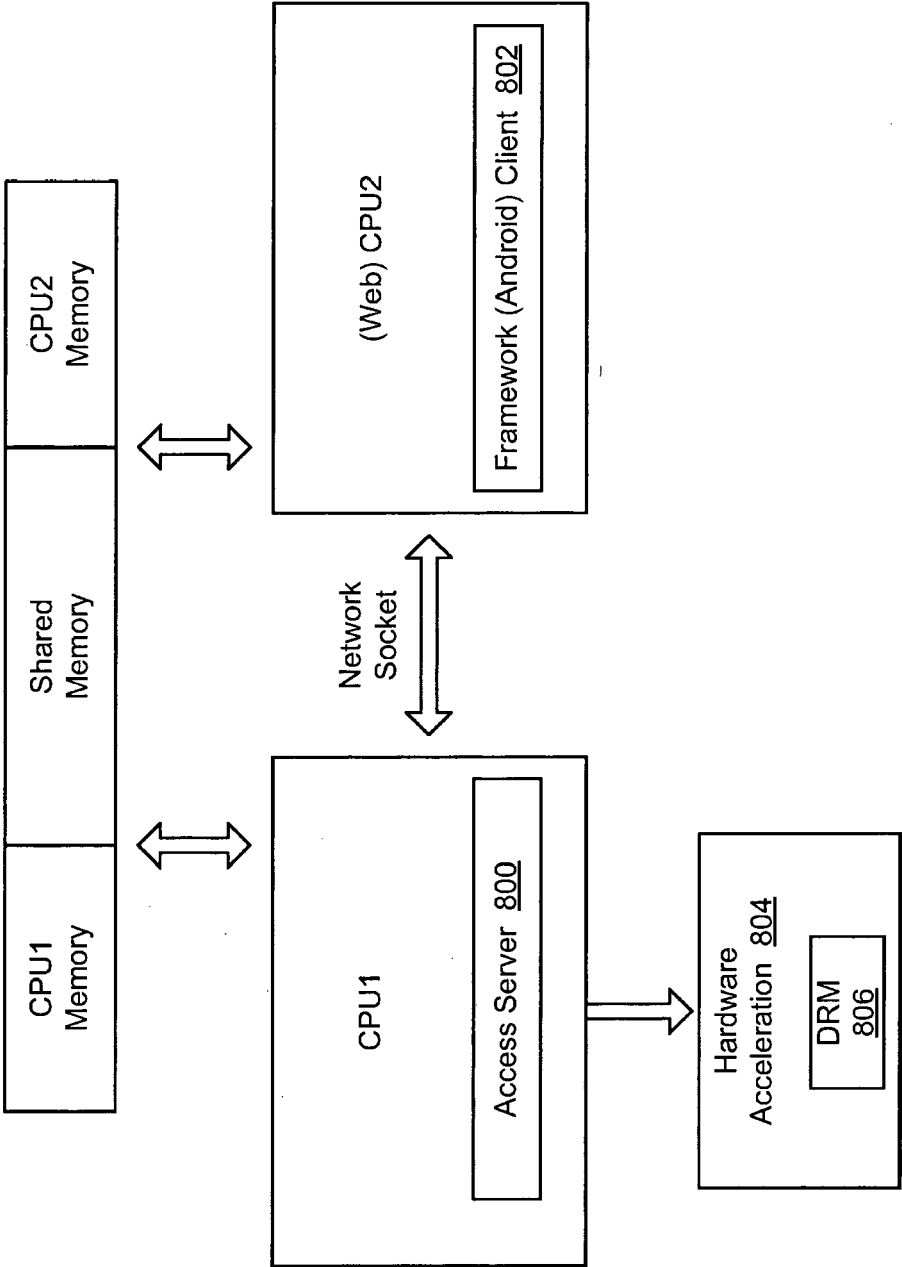


FIG. 8

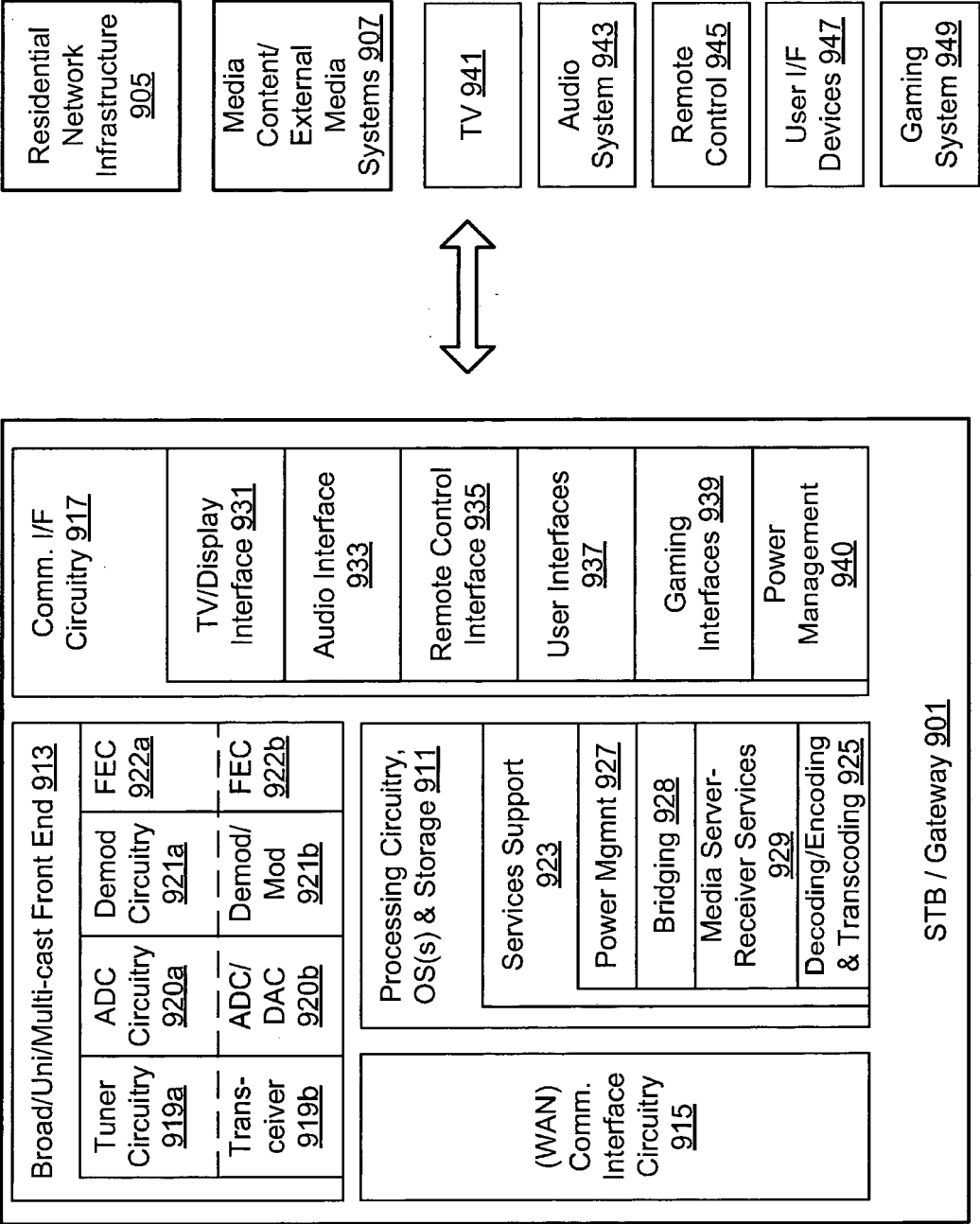


FIG. 9

1000

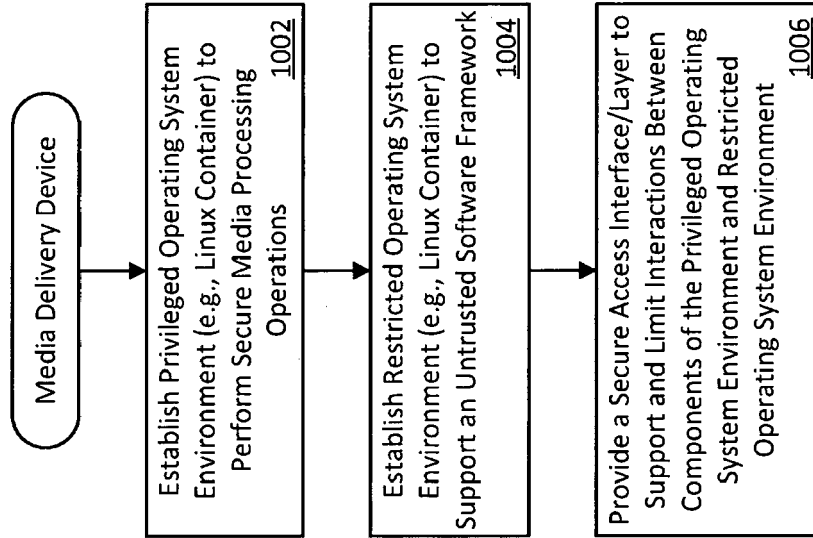


FIG. 10



EUROPEAN SEARCH REPORT

Application Number
EP 13 00 4995

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	WO 2006/011888 A1 (DISNEY ENTPR INC [US]; WATSON SCOTT [US]) 2 February 2006 (2006-02-02) * paragraph [0028] - paragraph [0049]; figures 1-3 *	1-15	INV. H04N21/443 H04N21/4627
X	EP 1 522 923 A2 (ST MICROELECTRONICS SA [FR]) 13 April 2005 (2005-04-13) * paragraph [0026] - paragraph [0040]; figures 2-5 *	1-15	
X	WO 2006/122024 A2 (MYRIO CORP [US]; DRAKE EDDIE [US]; ANNAPUREDDY SATISH [US]; WALD HANS) 16 November 2006 (2006-11-16) * paragraph [0024] - paragraph [0082]; figures 2-5 *	1-15	
X	US 2008/127355 A1 (LORCH JACOB R [US] ET AL) 29 May 2008 (2008-05-29) * paragraph [0034] - paragraph [0038]; figure 4 *	1-15	
X	US 2008/005171 A1 (CAMERON DONALD F [US]) 3 January 2008 (2008-01-03) * figure 2 *	1-15	
The present search report has been drawn up for all claims			TECHNICAL FIELDS SEARCHED (IPC)
			H04N
Place of search		Date of completion of the search	Examiner
Munich		19 December 2013	Marzal-Abarca, X
<p>CATEGORY OF CITED DOCUMENTS</p> <p>X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document</p> <p>T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document</p>			

2
EPO FORM 1503 03/82 (P04/C01)

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 13 00 4995

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

19-12-2013

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2006011888 A1	02-02-2006	NONE	
EP 1522923 A2	13-04-2005	EP 1522923 A2	13-04-2005
		US 2005081020 A1	14-04-2005
WO 2006122024 A2	16-11-2006	NONE	
US 2008127355 A1	29-05-2008	NONE	
US 2008005171 A1	03-01-2008	CN 101479715 A	08-07-2009
		CN 102682081 A	19-09-2012
		DE 112007001417 T5	07-05-2009
		US 2008005171 A1	03-01-2008
		US 2010169461 A1	01-07-2010
		US 2011125878 A1	26-05-2011
		WO 2008005720 A1	10-01-2008

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 61715785 A [0001]
- US 61725964 A [0001]
- US 61725790 A [0001]